UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/395,207	09/14/1999	SUNSHIN AN	K-105	5612

34610 7590 01/29/2004

FLESHNER & KIM, LLP
P.O. BOX 221200
CHANTILLY, VA 20153

EXAMINER	
WON, YOUNG N	

ART UNIT	PAPER NUMBER
2155	18

DATE MAILED: 01/29/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/395,207	AN ET AL.	
	Examiner	Art Unit	
	Young N Won	2155	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *20 November 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-15,17 and 18* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-15,17 and 18* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. §§ 119 and 120

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

13)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application) since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

a) ☐ The translation of the foreign language provisional application has been received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____.
4) ☐ Interview Summary (PTO-413) Paper No(s). _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

1. In view of the Appeal Brief filed on November 20, 2003, PROSECUTION IS HEREBY REOPENED. New grounds of rejection are set forth below.

To avoid abandonment of the application, appellant must exercise one of the following two options:

(1) file a reply under 37 CFR 1.111 (if this Office action is non-final) or a reply under 37 CFR 1.113 (if this Office action is final); or,

(2) request reinstatement of the appeal.

If reinstatement of the appeal is requested, such request must be accompanied by a supplemental appeal brief, but no new amendments, affidavits (37 CFR 1.130, 1.131 or 1.132) or other evidence are permitted. See 37 CFR 1.193(b)(2).

2. Claims 1-15, 17, and 18 are pending with this action.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-14, 15, and 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yata et al. ("ATM Transport Network Operation System Based on Object Oriented Technologies", NTT Transmission Systems Laboratories, IEEE 1994 pgs.838-842) in view of Okamura et al. (US 6636964 B1).

Independent:

As per claim 1, Yata teaches of a network management system (see abstract), comprising: a management system kernel that provides management systems with a run-time environment (see pg.838, Fig.1; abstract; and pg.839, section 2.2. TNMS Kernel); and a managed object generation environment that provides a development environment for managing applications (see pg.838, Fig.1 and section 2.1.1. GDMO Editor), wherein the management system kernel can at least one of dynamically add and dynamically modify managed object (MO) information (see pg.838, Fig.1 and pg.839 section 2.2. TNMS Kernel: "is a software library that realizes the functions needed for transport network OpS application... The structure allows represent calling and called relationships between library modules."). Yata does not explicitly teach that the management system kernel adds or updates MO information based upon an external meta file (EMM) from the managed object generation environment without interrupting an operation of the network management system. Okamura teaches of a management system kernel (see col.10, lines 19-23) adds or updates MO information

based upon an external meta file (EMM) (see col.7, lines 3-42) from the managed object generation environment without interrupting an operation of the network management system (see col.1, lines 29-35 and col.17, lines 3-7). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Okamura within the system of Yata by implementing external meta files to update objects without interrupting network management system operation because Yata discusses the basic requirements are resource and duration should be as small and as short as possible (performance) (see Yata: pg.838, first column, lines 35-36), and clearly with the improvement taught by Okamura, not only is performance greatly improved by negating the time lost by closing application programs and then restarting the operating system, but allowing sharing of same services by a plurality of execution environments provides flexibility to the system (see col.2, line 53 to col.3, line 5). Furthermore, Yata teaches that since shared management is needed due to the fact that each managed object is independent of the agent system, "a managed object location transparency mechanism was adopted" (see pg.840, section 3.3.4. Managed object location transparency).

As per claim 8, Yata teaches of a network management method comprising: (a) storing a dynamic class loading routine in a management system kernel (see pg.839, section 2.2.3. Data Bases Access (DBA) API); (b) initializing a managed system by constructing a managed object framework of the management system kernel that contains information of managed object (MO) classes (see pg.838-839, section 2.1.2. GDMO Translator); (c) creating MO instances and registering the MO instances in a

containment tree of the management system kernel according to the information of MO classes (see pg.839, section 2.1.3. Common Management Information Editor); (d) checking whether a dynamic class loading flag is on (Note: it is inherent when Yada teaches of "describing new definitions", "newly defined managed objects" or loading the managed objects from a plurality of databases, there is a trigger mechanism to inform the system whether the definition exists internally or must be retrieved externally), when receiving a management operation request from a management system (see pg.839, section 2.1.3. Common Management Information Editor: "debugger for application programs within the OpS... handles attribute classes generated by the GDMO translator"); and (e) updating MO information on the management system kernel (see pg.838, Fig.1 and pg.839 section 2.2. TNMS Kernel: "is a software library that realizes the functions needed for transport network OpS application... The structure allows represent calling and called relationships between library modules."). Yata does not explicitly teach of updating the MO information without interrupting an operation of the network management system by, waiting for all threads to complete execution, loading a dynamic library to the managed object framework utilizing the dynamic class loading routine when the dynamic class loading flag is on, and resetting the dynamic class loading flag to off. Okamura teaches of updating the MO information without interrupting an operation of the network management system (see col.1, lines 29-35 and col.17, lines 3-7) by, waiting for all threads to complete execution (see Fig.6 & Fig.7; col.3, line 65 to col.4, line 24; and col.12, line 36 to col.13, line 30), loading a dynamic library to the managed object framework utilizing the dynamic class loading routine

when the dynamic class loading flag (see Note above) is on, and resetting the dynamic class loading flag to off (see col.9, lines 65-67). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Okamura within the system of Yata by implementing steps mentioned above to avoid interrupting an operation of the network management system because Yata discusses the basic requirements are resource and duration should be as small and as short as possible (performance) (see Yata: pg.838, first column, lines 35-36), and clearly with the improvement taught by Okamura, not only is performance greatly improved by negating the time lost by closing application programs and then restarting the operating system, but allowing sharing of same services by a plurality of execution environments provides flexibility to the system (see col.2, line 53 to col.3, line 5). Furthermore, Yata teaches that since shared management is needed due to the fact that each managed object is independent of the agent system, "a managed object location transparency mechanism was adopted" (see pg.840, section 3.3.4. Managed object location transparency).

As per claim 15, Yata teaches a network management method, comprising: storing a dynamic class loading routine in a management system kernel of the managed system (see pg.839, section 2.2.3. Data Bases Access (DBA) API); updating the management system kernel by modifying managed object (MO) information in the management system kernel by utilizing the dynamic class loading routine (see pg.838, Fig.1 and pg.839 section 2.2. TNMS Kernel: "is a software library that realizes the functions needed for transport network OpS application... The structure allows represent calling and called relationships between library modules."); and generating the MO

information to be modified (see pg.839, section 2.1.3. Common Management Information Editor). Yata does not explicitly teach of generating a external meta file (EMM) in a managed object generation environment of the managed system wherein the dynamic class loading routine opens the EMM file to modify the MO information in the management system kernel (see col.7, lines 3-42 and col.10, lines 19-23). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Okamura within the system of Yata by implementing generating a external meta file (EMM) in a managed object generation environment of the managed system wherein the dynamic class loading routine opens the EMM file to modify the MO information in the management system kernel because Yata teaches that since shared management is needed due to the fact that each managed object is independent of the agent system, "a managed object location transparency mechanism was adopted" (see pg.840, section 3.3.4. Managed object location transparency), therefore, since Okamura teaches of sharing objects (see col.2, lines 45-47), one of ordinary skill in the art would employ the transparency mechanism of Okamura. Furthermore, Yata teaches, "behavior clauses of each managed object class definition were extended by adding C++ programs using TNMS Kernel's libraries" (see Yata: pg.841, lines 13-15).

Dependent:

As per claim 2, Yata further teaches wherein the management system kernel comprises: a communication module that provides communication with a network

manager (see); a managed object framework that maintains information on MO classes (see pg.838-839, section 2.1.2. GDMO Translator); a kernel that stores a dynamic class loading module and initializes the network management system (see pg.839, section 2.2. TNMS Kernel), wherein said kernel establishes an association with other management systems through the communication module (see pg.839, section 2.2. TNMS Kernel and pg.839, section 2.2.3. Data Bases Access (DBA) API), performs management operations on MOs, adds the MO information in the managed object framework using the dynamic class loading module and modifies the MO information in the managed object framework using the dynamic class loading module (see pg.839, section 2.2.4. MIB API); and a containment that organizes MO instances according to the information on MO classes and allows access to the MO instances when a management operation is performed in the network managed system (see pg.839, first column, lines 1-3 and lines 5-9).

As per claim 3, Yata further teaches wherein managed object framework maintains information on MO classes (see pg.838, section 2.1.1. GDMO Editor and pg.839, first column, lines 1-3 & 7-9) by registering MO class codes on a class information table (inherency).

As per claim 4, Yata further teaches wherein the kernel creates at least one dedicated agent to perform subsequent management operations from management systems with which an association has been established (see pg.839, lines 8-11).

As per claim 5, Yata further teaches wherein the managed object generation environment comprises: a MO compiler that compiles a MO script to generate the EMM

file and MO class codes (see pg.838-839, section 2.1.2. GDMO Translator); and a dynamic library storing the MO class codes (see pg.838, Fig.1 and pg.839 section 2.2. TNMS Kernel: "is a software library that realizes the functions needed for transport network OpS application... The structure allows represent calling and called relationships between library modules.").

As per claim 6, Yata further teaches wherein the EMM file includes MO class definition described in the MO script (see pg.839, lines 7-9), and identifies a location and name in the dynamic library of a corresponding MO class (inherent).

As per claims 7, Although Yata further teaches wherein the MO class codes are compiled and stored in the dynamic library, he does not explicitly teach that the library is a dynamic link library. Okamura teaches of dynamic link library (see col.9, lines 65-67). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Okamura within the system of Yata by implementing dll's within the network management system because dll's are well-known industry standard and employing such library makes the system more cost effective and increases usability and overcomes some of the limitations taught by Yata (see pg.838, first column, lines 24-30).

As per claim 9, Yata further teaches (f) performing the requested management operation and sending a management operation result to the management system requesting the management operation (see pg.840, lines 13-14) when the dynamic class loading flag is not on (see claim 8 rejection above).

As per claim 10, Yata and Okamura further teach wherein dynamic class loading routine of (e) comprises: opening an EMM file stored outside the management system kernel (see); and loading the dynamic library indicated by the EMM file (see claim 1 rejection above).

As per claim 11, Yata teaches of further comprising storing information about the management system requesting a management operation (see pg.838, Fig.1: "Existing GDMO Definitions" and pg.839, section 2.2.3. Data Base Access (DBA) API).

As per claim 12, Yata does not explicitly teaches of further comprising checking whether an additional thread can be created; creating a dedicated agent to take charge of subsequent management operations from the management system requesting an association if an additional thread can be created; and executing the dedicated agent thread and delivering association and management operation information to the dedicated agent to be utilized in interacting with the management system. Okamura teaches of checking whether an additional thread can be created; creating a dedicated agent to take charge of subsequent management operations from the management system requesting an association if an additional thread can be created (see col.7, line 58: "thread manager"); and executing the dedicated agent thread and delivering association and management operation information to the dedicated agent to be utilized in interacting with the management system (see col.8, lines 11-13). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Okamura within the system of Yata by implementing creation of threads and assigning agents to threads within the network management system

because Yata teaches of an Event Handling API that "controls multiple execution threads and invokes an event handling module on a newly created thread" (see pg.839, first column, lines 19-20) and that the thread scheme "realizes multiple managed object access in different agent systems" (see pg.839, second column, lines 13-15), therefore, one of ordinary skill in the art would employ multiple threads with dedicated agents.

As per claim 13, Yata further teaches wherein the dynamic class loading flag is set on when the management system requesting a management operation invokes the dynamic class loading function to perform one of adding and modifying MO information in the management system kernel (see claim 8 rejection above).

As per claim 14, Yata further teaches wherein the management system requesting the management operation invokes the dynamic class loading function by sending a control signal (inherent).

As per claim 17, Yata and Okamura further teach wherein the MO information to be modified is stored in the managed object generation environment in the form of a dynamic link library (see claim 7 rejection above).

4. Claim 18 is rejected under 35 U.S.C. 103(a) as being unpatentable over Yata et al. ("ATM Transport Network Operation System Based on Object Oriented Technologies", NTT Transmission Systems Laboratories, IEEE 1994 pgs.838-842) and Okamura et al. (US 6636964 B1) and further in view of Draaijer et al (US 5987463 A).

As per claim 18, Yata and Okamura teaches all the limitation including wherein the MO information is modified in the management system kernel (see pg.841, lines 12-15). Yata and Okamura do not explicitly teach wherein the EMM indicates an address of a dynamic link library corresponding to the MO information to be modified. Draaijer teach wherein the EMM indicates an address of a dynamic link library corresponding to the MO information (see col.10, lines 19-27). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Draaijer within the system of Yata and Okamura by implementing meta data to address corresponding MO dll's within the network management system because Draaijer teaches that external procedures from external databases (as taught by Yata) cannot describe themselves and therefore must be dynamically "linked in" (see col.10, lines 28-34).

5. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Young N Won whose telephone number is 703-605-4241. The examiner can normally be reached on M-Th: 8AM-6PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Hosain T Alam can be reached on 703-308-6662. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is 703-305-3900.

Young N Won

January 14, 2004

HOSAIN ALAM
SUPERVISORY PATENT EXAMINER